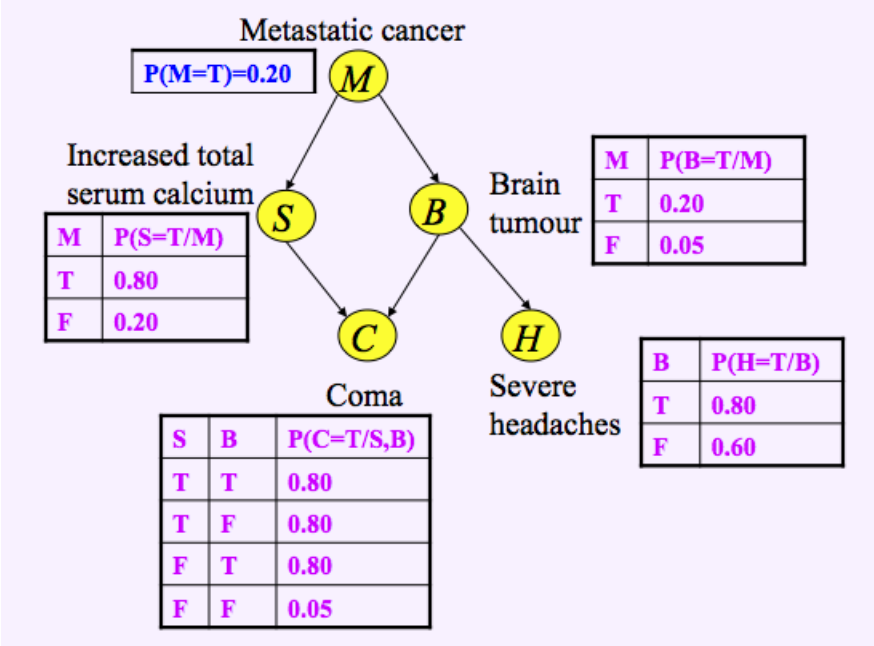
Delivery 2

Daniel Salgado Rojo

Data Visualization and Modelling. Course 2017/2018.

5) Consider Figure 1 and do the following:



M	P(S=T/M)
T	0.80
F	0.20

M	P(B=T/M)
T	0.20
F	0.05

B	P(H=T/B)
T	0.80
F	0.60

S	B	P(C=T/S,B)
T	T	0.80
T	F	0.80
F	T	0.80
F	F	0.05

Figure 1: Bayesian network of study and conditional probability tables.

(a) Using **gRain** develop the corresponding Bayesian Network and use it to compute $P(M = T/H = F)$ and $P(M = F/H = T)$.

(b) Develop the LS Algorithm to find these two probabilities with R and compare the results with previous item.

(c) Develop the LW Algorithm to find these two probabilities with R and compare the results with previous items.

(d) Compute exactly "by hand" $P(M = T = H = F)$, and compare with (a), (b) and (c).

(e) For the evidence "H = F" and the query variable M, compute the Kullback-Leibler divergence for the LS Algorithm, and also compute it for the LW Algorithm, and compare them. Which algorithm seems to be better?

Solution. (a)

With the following block of R-code we build the Bayesian Network from Figure 1, in particular the tables of probabilities of each node.

```
library(gRain)
library(gRbase)
library(Rgraphviz)
library(RBGL)

par(mfrow = c(1,1))
tf <- c("true","false")

# Specify the CPTs:
node.M <- cptable(~ M, values = c(2, 8), levels = tf)
node.S <- cptable(~ S|M, values = c(8, 2,2,8), levels = tf)
node.B <- cptable(~ B|M, values = c(2, 8, 5, 95 ), levels =
   tf)
node.C <- cptable(~ C|S+B , values = c(8, 2, 8, 2,8,2,5,95 ),
    levels = tf)
node.H <- cptable(~ H|B, values = c(8, 2, 6, 4 ), levels = tf
   )

# Create an intermediate representation of the CPTs:
nodes <- list(node.M,node.S,node.B,node.C, node.H)
plist <- compileCPT(nodes)

# Create a network of name "Norman.net", for instance:
BN<-grain(plist)
summary(BN)
# The graph:
BN.plot=plot(BN)
BN.plot
```

We also obtain a graphical representation (see Figure 2) of the obtained BN to check that it has been constructed correctly and coincides with that from Figure 1.

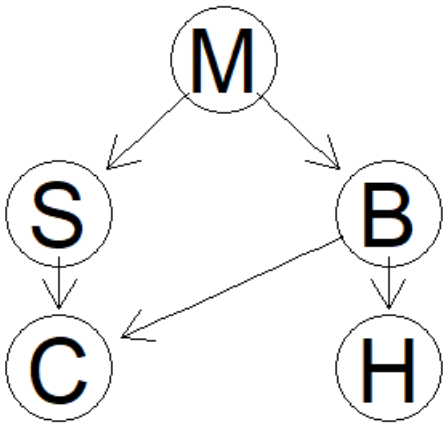


Figure 2: DAG associated to the Bayesian Network of study.

In order to compute the probabilities $P(M = T/H = F)$ and $P(M = F/H = T)$ we consider the following piece of code:

```
BNa1<-setEvidence(BN,nodes=c("H"),
                  states=c("false"))
marg1=querygrain(BNa1,nodes
                 =c("M"), type="marginal")

#P(M=T/H=F)= 0.1875
marg1$M["true"]


BNa2<-setEvidence(BN,nodes=c("H"),
                   states=c("true"))
marg2=querygrain(BNa2,nodes
                 =c("M"), type="marginal")

#P(M=F/H=T)= 0.7922078
marg2$M["false"]
```

Thus, the results are:

- $P(M = T/H = F) = 0.1875$
- $P(M = F/H = T) = 0.7922078$

◀

In order to do the next two sub-exercises we will use the following function:

```
binary_random <- function(p.prior.true){
    seed <- runif(1)
    if(seed < p.prior.true){return("true");}
    else{return("false");}
  }
```

For the sub-exercises (b) and (c) we fix a total number of iterations

```
run_iterations = 10^6
```

to compute the approximated probabilities by using LS and LW methods, respectively.

Solution. (b)

```
  #LS Algorithm for a single evidence and the particular
  #BN from the current exercise.
  LS_alg <- function(pred.node = "M",
                     evid.node = "H",
                     pred.value = "true",
                     evid.value = "false",
                     iterations = run_iterations){

    count.pred_evid = 0; count.evid = 0;

    iter_counter = 0
    while(iter_counter < iterations){

      #Value assigntion
      M.value <- binary_random(plist$"M"[["true"]])
      S.value <- binary_random(plist$"S"[ "true" , M.value])
      B.value <- binary_random(plist$"B"[ "true" , M.value])
      C.value <- binary_random(plist$"C"[,, B.value]["true",
   S.value])
      H.value <- binary_random(plist$"H"["true", B.value])

      #Update counters
      node_value <- function(node = "M"){
        if(node == "M"){return(M.value)}
        else if(node == "S"){return(S.value)}
        else if(node == "B"){return(B.value)}
        else if(node == "C"){return(C.value)}
        else if(node == "H"){return(H.value)}
        else{print("Node not found");return(NA);}
```

```
      }

      e_observed <- node_value(evid.node)
      x_observed <- node_value(pred.node)

      if(e_observed == evid.value){
        count.evid = count.evid +1;
        if(x_observed == pred.value){
          count.pred_evid = count.pred_evid +1}
        }
      iter_counter = iter_counter +1;
    }#main simulation while

    return(count.pred_evid/count.evid)
  }#end function

  #P(M=T/H=F)
  estimated_lsprob1 <- LS_alg("M","H","true", "false")
  #P(M=F/H=T)
  estimated_lsprob2 <- LS_alg("M","H","false", "true")
```

The results of a single run are:

- $P(M = T/H = F) = 0.1867083$
- $P(M = F/H = T) = 0.79238$

◀

Solution. (c)

```
#LW Algorithm for a single evidence and the particular
  #BN from the current exercise.
  LW_alg <- function(pred.node = "M",
                     evid.node = "H",
                     pred.value = "true",
                     evid.value = "false",
                     iterations = 10){

    count.pred_evid = 0; count.evid = 0;
    likelihood.e <-0

    iter_counter = 0
    while(iter_counter < iterations){

      #Value assigntion root nodes (M)
```

```
16      if("M" == evid.node){
          M.value <- evid.value
18        likelihood.e <- plist$"M"[[evid.value]]
        }
20      else{
          M.value <- binary_random(plist$"M"[["true"]])
22      }
        #Value assignation for childs
24      #S
        if("S" == evid.node){
26        S.value <- evid.value
          likelihood.e <- plist$"S"[ "true" , M.value]
28      }
        else{
30        S.value <- binary_random(plist$"S"[ "true" , M.value
    ])
        }
32      #B
        if("B" == evid.node){
34        B.value <- evid.value
          likelihood.e <- plist$"B"[ "true" , M.value]
36      }
        else{
38        B.value <- binary_random(plist$"B"[ "true" , M.value
    ])
        }
40      #C
        if("C" == evid.node){
42        C.value <- evid.value
          likelihood.e <- plist$"C"[,, B.value]["true", S.value
    ]
        }
44      else{
46        C.value <- binary_random(plist$"C"[,, B.value]["true"
    , S.value])
        }
48      #H
        if("H" == evid.node){
50        H.value <- evid.value
          likelihood.e <- plist$"H"["true", B.value]
52      }
        else{
54        H.value <- binary_random(plist$"H"["true", B.value])
        }
56
```

```
      #Update counters
      node_value <- function(node = "M"){
        if(node == "M"){return(M.value)}
        else if(node == "S"){return(S.value)}
        else if(node == "B"){return(B.value)}
        else if(node == "C"){return(C.value)}
        else if(node == "H"){return(H.value)}
        else{print("Node not found");return(NA);}
      }

      e_observed <- node_value(evid.node)
      x_observed <- node_value(pred.node)

      if(e_observed == evid.value){
        count.evid = count.evid + likelihood.e;
        if(x_observed == pred.value){
          count.pred_evid = count.pred_evid +likelihood.e}
      }
      iter_counter = iter_counter +1;
    }#main simulation while

    return(count.pred_evid/count.evid)
  }#end function


  #P(M=T/H=F)
  estimated_lwprob1 <- LW_alg("M","H","true", "false")
  #P(M=F/H=T)
  estimated_lwprob2 <- LW_alg("M","H","false", "true")
```

- $P(M = T/H = F) = 0.2080199$
- $P(M = F/H = T) = 0.7924663$

We see that for a million number of iterations the LS method seems to be better than the LW method since the estimated probabilities are much closer to the exact probabilities computed in (a). We will discuss more about this at the end of sub-exercise (d).

◀

Solution. (d)

We use the notation X_T to denote $X = T$ and X_F to denote $X = F$.

Applying Bayes' Theorem we have that

$$P(M_T/H_F) = \frac{P(H_F/M_T)P(M_T)}{P(H_F)} \tag{1}$$

Now,

1)

$$\begin{aligned} P(H_F/M_T) &= P(H_F/B_T, M_T)P(B_T/M_T) + P(H_F/B_F, M_T)P(B_F/M_T) \\ &= P(H_F/B_T)P(B_T/M_T) + P(H_F/B_F)P(B_F/M_T) \\ &= 0.2 \times 0.2 + 0.4 \times 0.8 = 0.36 \end{aligned}$$

where in the second equality we have used the Markov condition, that H is independent of M given its parents (which is B),

2) $P(M_T) = 0.2$,

3) $P(B_T) = P(B_T/M_T)P(M_T) + P(B_T/M_F)P(M_F) = 0.2 \times 0.2 + 0.05 \times 0.08 = 0.08$,

4) $P(H_F) = P(H_F/B_T)P(B_T) + P(H_F/B_F)P(B_F) = 0.2 \times 0.08 + 0.4 \times (1 - 0.0800) = 0.384$.

and with all these probabilities we have that

$$P(M_T/H_F) = \frac{P(H_F/M_T)P(M_T)}{P(H_F)} = \frac{0.36 \times 0.2}{0.384} = 0.1875 \tag{2}$$

Similarly we could calculate $P(M_F/H_T)$ and see that it coincides with the value obtained using **gRain**, $P(M_F/H_T) \approx 0.7922078$, (as it happens with $P(M_T/H_F) = 0.1875$).

Now we would like to compare these exact results with the results obtained using the LS and LW algorithms.

Consider the following array of integers

```
iter.vector <- c(10^2, 2*10^2, 3*10^2, 5*10^2, 10^3,4*10^3,10^4)
```

whose components are iteration numbers for the main loop of the LS and LW algorithms.

For each number of iterations in `iter.vector` we perform `reps` $\in \{20, 30\}$ LS and LW simulations to finally average and obtain more accurate estimations of the approximated probabilities $P(M_F/H_T)$ and $P(M_T/H_F)$. The results are shown in figures 3 and 4.

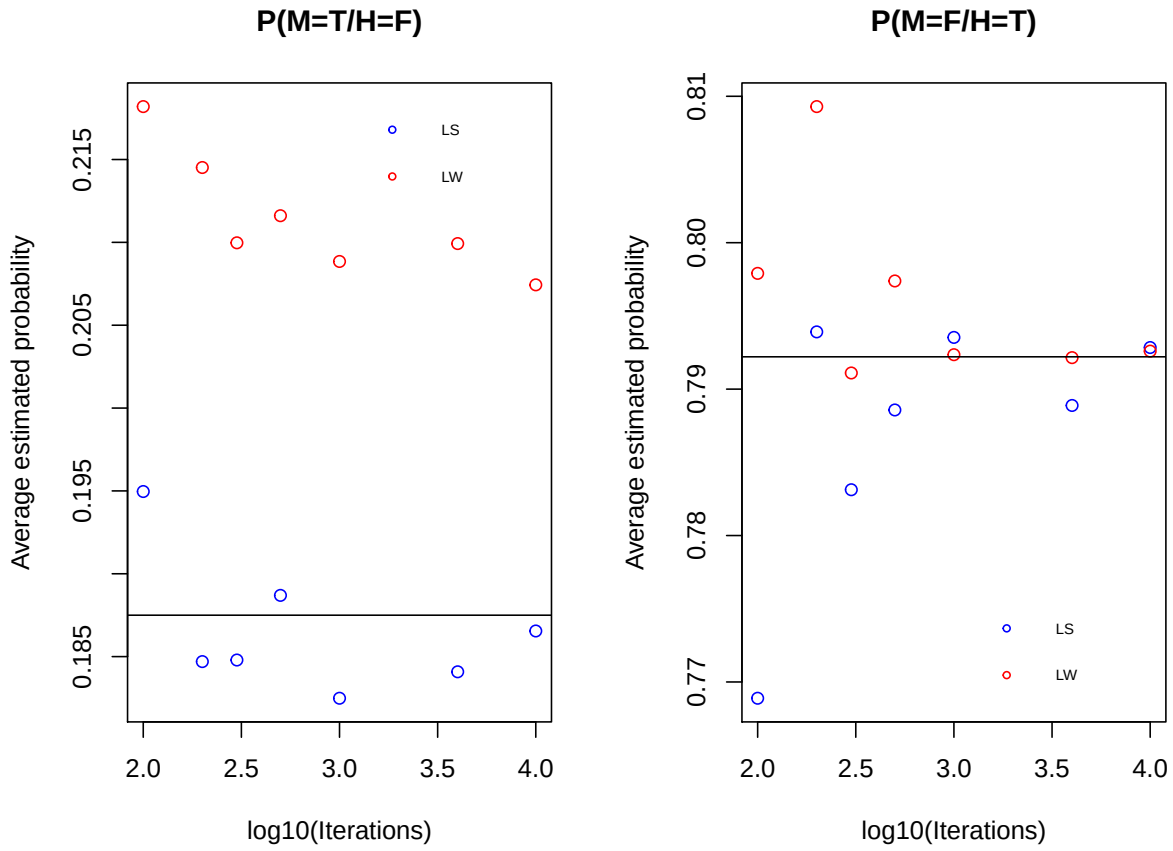


Figure 3: Averaged estimated probabilities with `reps` $= 20$ simulations and as a function of the number of iterations for both the LS and the LW algorithms.

By analysing both pictures we see that for the probability $P(M_F/H_T)$ both algorithms behave quite equally in terms of the convergence of the KL-Divergence to zero. However, for the probability $P(M_T/H_F)$, for some reason we see that LS converges to the exact value after about $5 \cdot 10^3$ iterations, whereas the LW method leads to probabilities quite far away from the exact value as the number of iterations increases.

With the obtained results we would conclude that the Algorithm LS is better than the LW, at least for the particular problem we are dealing with.

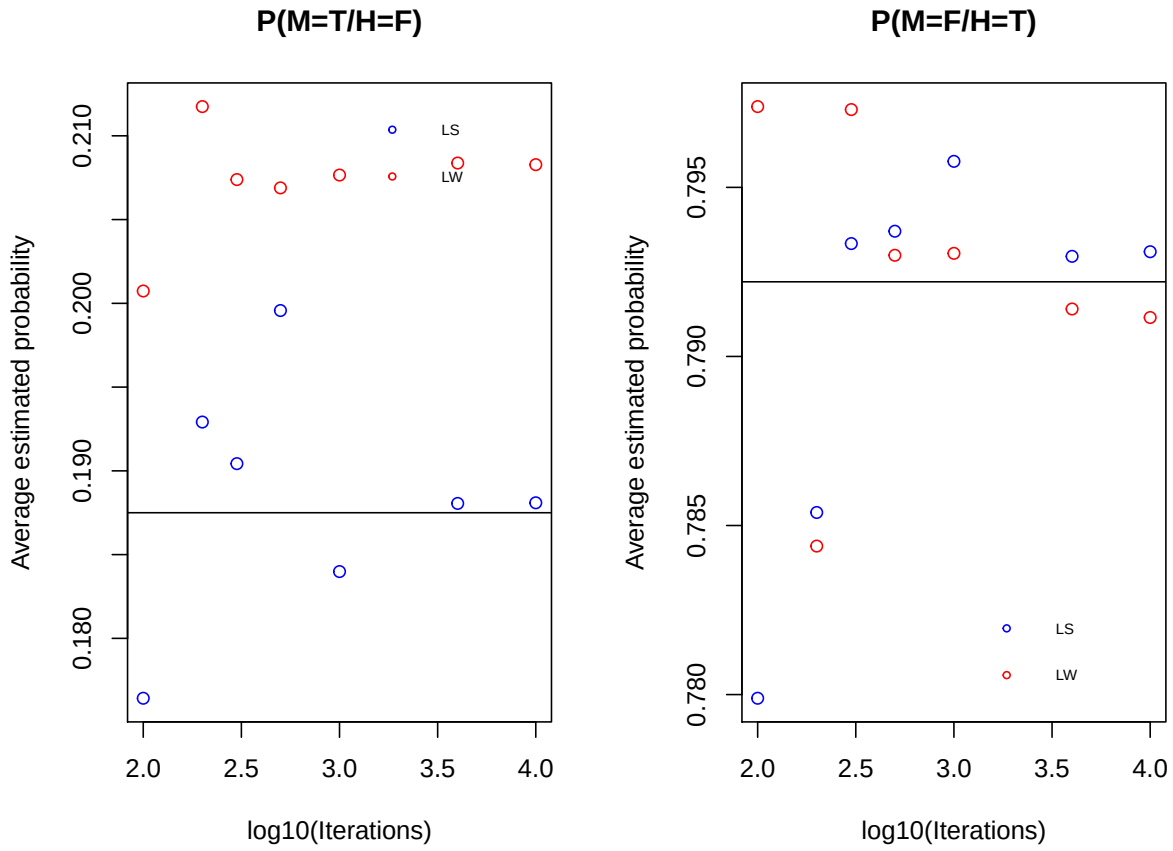


Figure 4: Averaged estimated probabilities with `reps` = 30 simulations and as a function of the number of iterations for both the LS and the LW algorithms.

◀

Solution. (e) Since we are working with binary random variables the calculation of the KL divergence is quite straight forward:

$$\begin{aligned}\mathrm{KL}\left(P(M/H=F), P'(M/H=F)\right) = P(M=T/H=F)\log\left(\frac{P(M=T/H=F)}{P'(M=T/H=F)}\right)\\ + P(M=F/H=F)\log\left(\frac{P(M=F/H=F)}{P'(M=F/H=F)}\right)\end{aligned}$$

where P' denotes the estimated/approximated probability distribution obtained by the LS or LW algorithms, in our case.

Similarly as we have done at the end of the sub-exercise (d), we have performed `reps` = 30 simulations with all the iteration numbers from the vector `iter.vector` in order to average the 30 KL values obtained (for the LS and the LW algorithms), for each iteration number, and then present the results graphically in Figure 5.

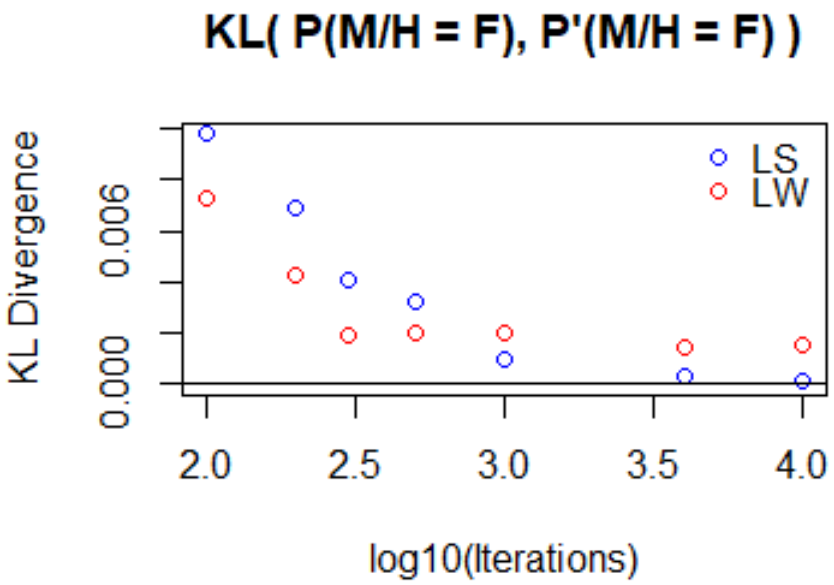


Figure 5: Averaged KL divergence with `reps` = 20 simulations and as a function of the number of iterations for both the LS and the LW algorithms.

From the KL divergences obtained using both methods, we arrive to similar conclusions as the ones we commented for the estimated probabilities $P(M = T/H = F)$:

- We see that the LS algorithm practically converges to 0 after about 5×10^3 iterations.
- On the contrary, the LW algorithm seems to approach to zero for smaller iteration numbers but then seems to remain constant or tend to zero very slowly as the number of iterations increases.

Taking into account the results obtained in the sub-exercise (d) and the ones we have just commented, we would say that for the current problem the LS algorithm is better than the LW algorithm.

It would be interesting to see what happens in a case where some of the exact probabilities we want to estimate are very close to zero, since then the LS is expected to perform worse. ◀

References

[1] Bayesian Networks Slides, Rosario Delgado, UAB. Available in [2].

[2] GitHub repository `https://github.com/dsalgador/Bayesian-Networks-Course`.